Frequency response of fractional differencing and integration

- Informal introduction, showing some motivations on why the fractional difference is an interesting **metric** on a price series.

- Most references talk about the **time domain** behavior of fractional differencing, so we thought we would show some examples of frequency domain behavior.

- These are companion slides to the github repo at:

  - https://github.com/diffent/fracdiff

December 22, 2019
diffent.com

Intro

- From other authors*, we know that differencing and fractional differencing are useful filters to apply to econometric (price) series data to a) make such data stationary, and b) to try to retain more of the long term behavior of the series for further analysis.

- We also show here an interpretation of a fracdiff as a single number metric to use to show how a price has changed, not just from yesterday, but from all prior days.

- Fractional differencing is a type of digital filter, so it is useful to consider the **frequency characteristics** of this filter to see what we are doing to our data in the frequency domain, and to give ideas about other types of similar filters that may be applied to achieve similar results (e.g. stationarity while retaining some long term memory).

*** Some intro-level, but detailed discussions:**

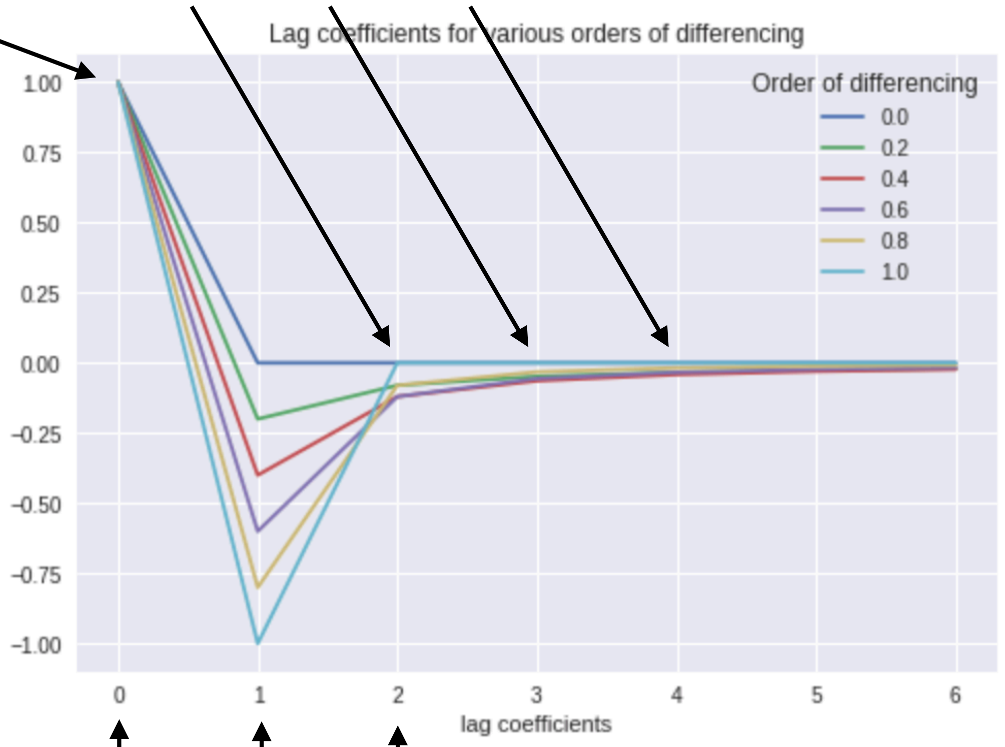
http://kidquant.blogspot.com/2019/02/welcome-and-introduction-to-fractional.html

https://towardsdatascience.com/preserving-memory-in-stationary-time-series-6842f7581800

Motivation:
When we compute an ordinary **daily difference (in prep for computing daily return as %), we** only **take the difference between** today **and** yesterday **(e.g. difference of closing price)**

weight on today price

ordinary difference: all weights on prices older than yesterday are zero



ordinary differencing (e.g. used in daily return %) throws away a lot of information

....

"on purpose!"

weight on yesterday price for ordinary difference d = 1

today

yesterday

day before yesterday

even older days...

Chart represents the weights to apply to today and prior prices in a price time series to compute a fractional difference at various "order of differencing" levels. This method can apply to any time series, not just price, but we use price as a well understood example (e.g. percent daily change in price is often important).

Chart from: https://towardsdatascience.com/preserving-memory-in-stationary-time-series-6842f7581800

"You threw away the rest of the data....on PURPOSE!"

Maybe we shouldn't throw away so much data when computing a metric...?

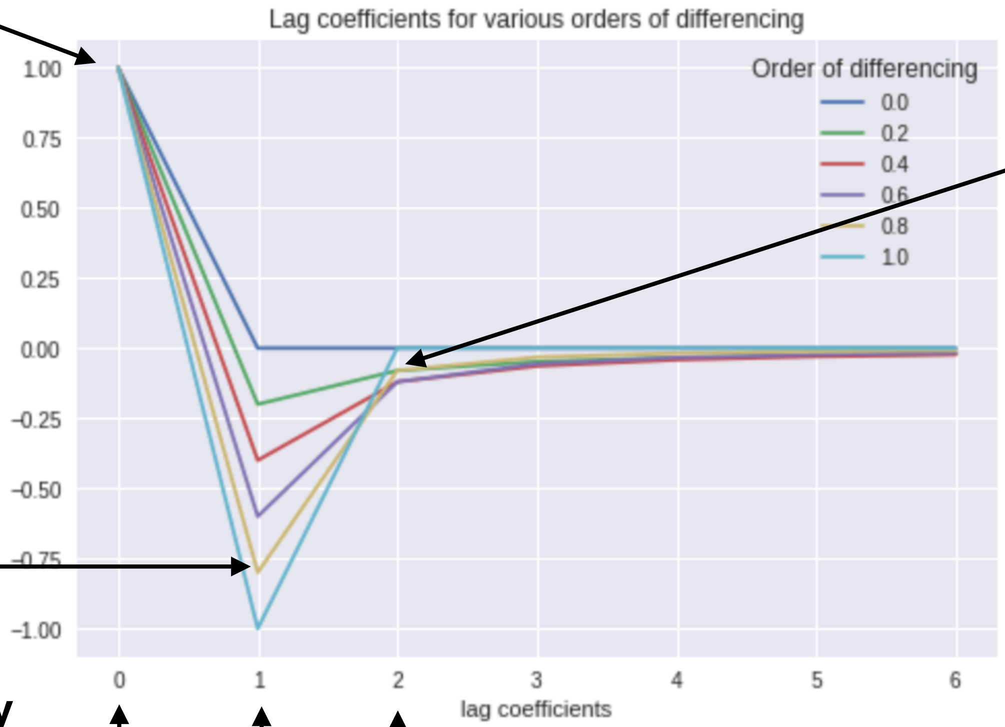
- Ordinary daily return metric: throws away data "before yesterday"

- Fractional difference return metric: accounts for not only yesterday, but all days prior...keeps older data, to some extent

Colloquial reasoning:
When we compute a fractional **difference,**
we take a weighted sum **of the (current price + smaller weight (negative) on yesterday's price, + even smaller weight (negative) on day-before-yesterday price, etc...)**

weight on today price always **1.0 for all differencing orders (d)**

take example d = 0.8



Then we take 8% of the day before yesterday price **and subtract that**

weight on yesterday price for FRACTIONAL difference d = 0.8

Etc... for even older days

note weight on yesterday is not -1 any more... we only take 80% of "yesterday's" price and subtract it

today

yesterday

day before yesterday

Older weights are always negative and keep getting closer to zero

Chart from: https://towardsdatascience.com/preserving-memory-in-stationary-time-series-6842f7581800

Exact weights for fractional difference d = 0.8 example

```
w[0] = 1.000000       weight on today price, the only positive weight
w[1] = -0.800000      weight on yesterday price (negative)
w[2] = -0.080000      weight on day before yesterday price (negative, much smaller)
w[3] = -0.032000      weight on one more day back price (negative, much smaller still)
w[4] = -0.017600      etc...
w[5] = -0.011264
w[6] = -0.007885
w[7] = -0.005857
w[8] = -0.004539
w[9] = -0.003632
```

Aside from the somewhat complicated technical details about "stationarity of time series" and "Augmented Dickey-Fuller tests"...

Click here if you dare! --> https://en.wikipedia.org/wiki/Augmented_Dickey–Fuller_test

...we see that a fractional difference **is ONE NUMBER, a "metric" to see how a price has changed from "today" versus** all **prior days,**

giving much more weight to more recent days **in that computation**

In other words:

- Ordinary difference: shows change versus yesterday.
- Fractional difference: shows change versus yesterday, but not as much, and also includes days before yesterday, with increasingly lesser influence as we go back in time.

How to give more weight to older data? set d lower

For d = 0.8 (closer to an ordinary daily difference)

```
w[0] = 1.000000       weight on today price, the only positive weight
w[1] = −0.800000      weight on yesterday price (negative)
w[2] = −0.080000      weight on day before yesterday price (negative, much smaller)
w[3] = −0.032000      weight on one more day back price (negative, much smaller still)
w[4] = −0.017600      etc...
w[5] = −0.011264
w[6] = −0.007885
w[7] = −0.005857
w[8] = −0.004539
w[9] = −0.003632
```

For d = 0.5 (giving less weight to yesterday versus above 0.8 case, and more weight to prior days)

```
w[0] = 1.000000   today's weight is always 1.0 in the fracdiff formula
w[1] = −0.500000  yesterday's weight is now half, instead of 0.8
w[2] = −0.125000  day before yesterday's weight is higher than in 0.8 case above
w[3] = −0.062500  etc...older days now have more weight in the final metric
w[4] = −0.039062
w[5] = −0.027344
w[6] = −0.020508
w[7] = −0.016113
w[8] = −0.013092
w[9] = −0.010910
```

--> As d **gets closer to zero, we give more weight to older data, and less weight to more recent data (in particular : less weight to "yesterday")**

Frequency response of fractional differencing and integration

- Fractional differencing is a type of digital filter.
- For fraction = 1.0, this resolves to ordinary differencing of adjacent-in-time values, e.g. for a price series P(t) the first difference at each point in time is: P(t)-P(t-1). Fraction = 2 is the 2nd derivative, etc.
- For fraction = 0 --> pass-thru (no change to original data).
- For fraction close to 0 but greater than 0, this suggests a small amount of "high pass" type of filtering (some low frequencies or slow changing behavior are attenuated, and higher frequencies or rapidly changing behaviors are retained or amplified).
- For fraction increasing toward 1.0, more of the low frequencies are attenuated as we increase the fraction.
- To get the inverse function, e.g. undo the fractional difference, reverse the fraction (d = -1*d) when computing the filter coefficients. This is fractional integration or fractional summing in the case of discrete time series data.
- Fractional **integration** is also a type of digital filter, but where **low frequencies are boosted** and **high frequencies are attenuated.**
- For fraction < 0, this suggests fractional integration. Fraction = -1 implies ordinary integration (1 level), Fraction = -2 is a double integral, etc.

Nice features of this filter

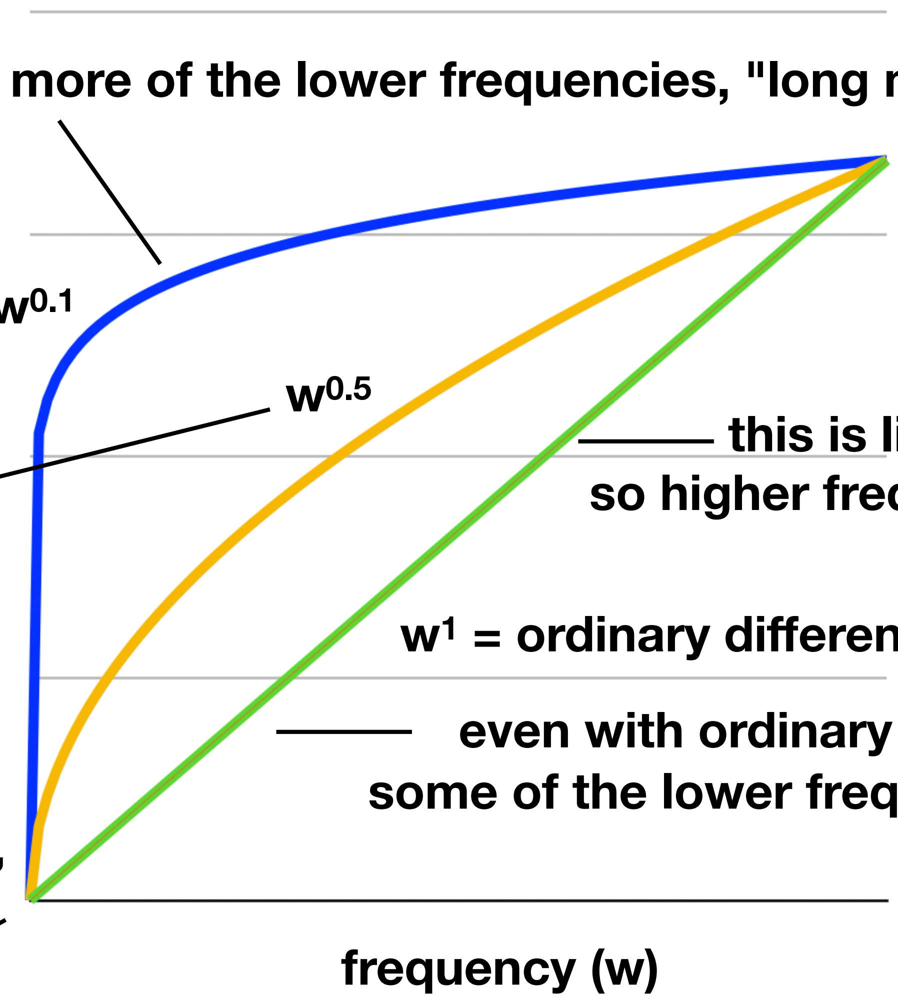
- To compute each value of the output, it is just a dot product of the input data from that point in time "and prior," dotted with the fracdiff weight vector. The fracdiff is only one of an infinite number of filters that can be applied using this method. Different weight vector = different filter.
- The weight vector is determined by the level of fractional differencing we want to do. For the simple case of ordinary differencing with the fraction d = 1, the weight vector is [1, -1], e.g. to compute one of the output values at t, we take the Price at t, multiply it by 1, then we take the next prior value, multiply it by -1, and sum all that together (the final value in the "dot" operation). So in the case of a price time series, we get output(t) = 1*P(t) + -1*P(t-1) for the simple difference, which of course resolves to the simple difference case P(t)-P(t-1)
- The references provided (and our code on github) show details of how the weight vector is computed for the general fracdiff case.
- Only the DC (constant, long term level) is totally removed. The other low frequencies are kept, albeit at a low level.
- This allows almost complete reversibility of this filter (which is unlike the case of lossy filters, where more frequency data is thrown away).
  - In fact, in our (and many) code implementations of fracdiff, the lost constant gets re-added back during integration such that the original series is recovered.
- In the traditional non-fractional case, if we difference a series, we can get the original series back exactly (up to some numerical noise) by re-integrating it.
  - Such is the method used to good results in the ARIMA (auto regressive integrated moving average) models:
  - https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/
  - The fractional differencing filter is itself an AR model of sorts, (auto regressive) since it only includes prior values of its own time series, with weights applied. It does not include forecasted results fed back into the model (the MA portion of ARIMA). For a contrast, see Gretl "filter" function: http://gretl.sourceforge.net/gretl-help/funcref.html#filter which does allow such feedback filters.
    - other systems have similar functions
  - ARFIMA is the ARIMA extension to fractional differencing (the added F stands for Fractional of course):
    - https://www.stata.com/features/overview/arfima/

For d > 0 = differencing, we get power-function **x^y frequency response**

disregarding phase shifts for this intro...
(shift of time series peaks)

frequency response of fractional differencing to level d = frequency ^{d}

This conceptual chart shows 3 separate fractional differencing frequency responses at d = 0.1, d=0.5, and d=1



there is nothing special about d=0.5 half derivative except that we call it "square root"...it is one in a family of filters that we can use to clean up our data

half derivative filter frequency response = sqrt(frequency) keeps more of the lower frequencies, & longer-term behavior

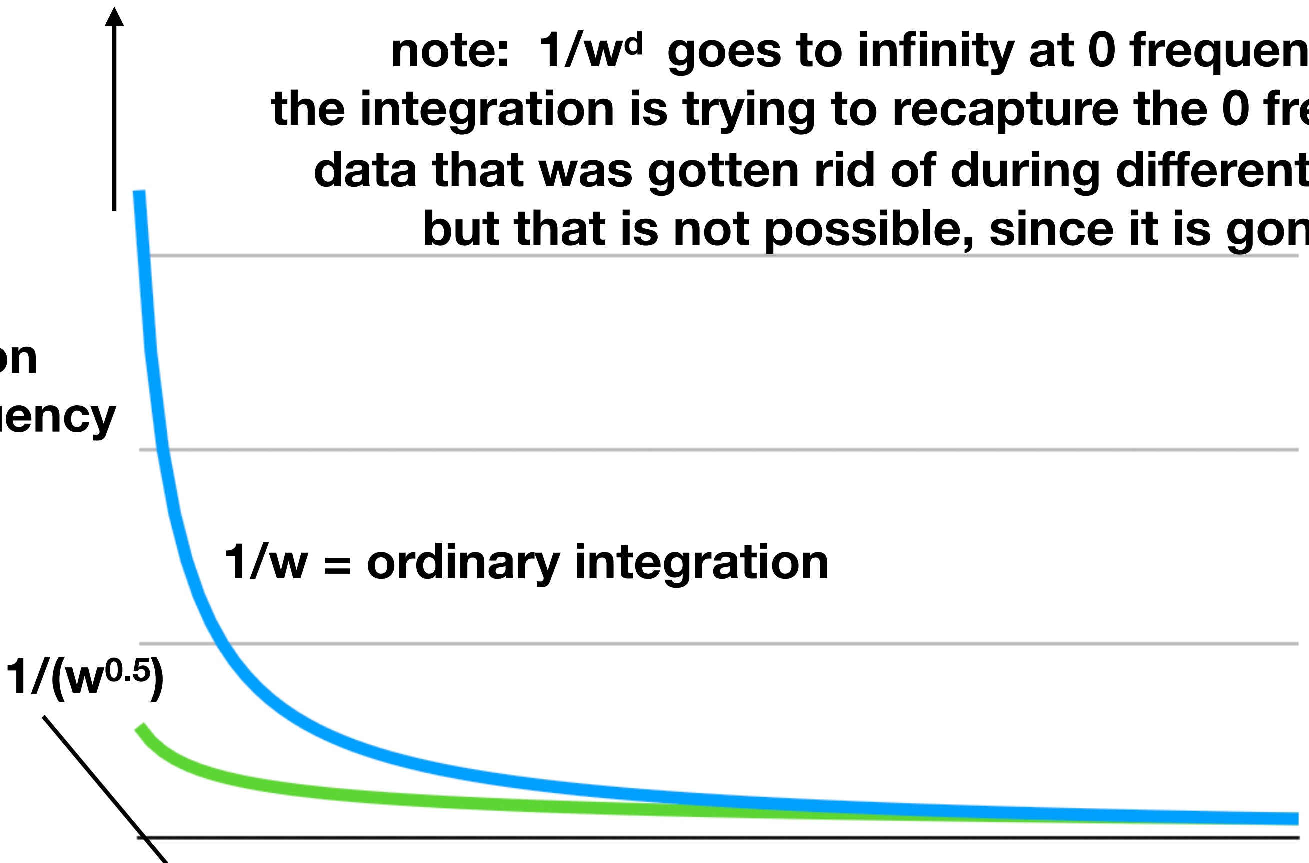
in all cases, the zero frequency (constant) is filtered out

If we set d close to zero, we get "hardly any" filtering... d = 0 returns the original time series

Fractional integration **frequency response examples**

Simple integration = hyperbolic function 1/x

note: low **frequencies get** magnified
high **frequencies get** attenuated
(the reverse of fractional differencing)



disregarding phase shifts for this intro... (shift of time series peaks) ...the imaginary unit j

time domain function is integration

$$\int_{-\infty}^{t} f(\tau)d\tau$$

$$\frac{F(\omega)}{j\omega} + \pi F(0)\delta(\omega)$$ **freq domain**

Where does this come from?

- Fourier transform theory, a large body of work.
- Note that even though time series in finance are not (usually or mostly) periodic, we can still apply filters which deal with the concepts of "frequency."
- Derivatives and integrals appear in the Fourier transform table (dealing with periodic functions); does this mean we cannot apply derivatives and integrals to non-periodic functions? No.
- Confusion sometimes comes in when we talk about "periods" or "90 degree phase shifts", etc. applied to non periodic time series like asset prices.
- We leave that out in this intro because we are aware that peak shifts in time occur when we apply many types of filters, we just have to be mindful of the fact that peaks shift, and we have to account for them.
- Purists will know that we really mean Discrete fourier transform theory, but this is good enough to start.

Quick intro to the theory side:

d>0 fractional difference
d=0 pass-thru filter no change
d<0 fractional integration

original time series

in frequency domain:

frequency to the n power

$$\frac{d^n f(t)}{dt^n} \quad \Bigg| \quad (j\omega)^n F(\omega)$$

difference it to level n

j is the imaginary unit, it indicates that differentiating or differencing a time series causes a shift in time series peaks ("phase shift" in frequency domain)

Fourier transform (FFT) of original time series

differencing in time domain is equivalent to this particular filter type in frequency domain: $(jw)^n$

How much shift: to be determined...

From this nice table of Fourier transforms:

https://ethz.ch/content/dam/ethz/special-interest/baug/ibk/structural-mechanics-dam/education/identmeth/fourier.pdf

Traditional Fourier transforms for diff and int compared

https://ethz.ch/content/dam/ethz/special-interest/baug/ibk/structural-mechanics-dam/education/identmeth/fourier.pdf

$$\frac{d^n f(t)}{dt^n} \quad \Big| \quad (j\omega)^n F(\omega)$$

differentiation

$$\int_{-\infty}^{t} f(\tau)d\tau \quad \Big| \quad \frac{F(\omega)}{j\omega} + \pi F(0)\delta(\omega)$$

integration

only one level of integration
is shown in the table
but we know we can integrate
twice to "undo" two derivatives

note that $(jw)^n = 1/(jw)^n$

some sort of constant for the theory guys
to recover the zero frequency that is completely
lost by differentiation: (jw) at 0 is 0 so
the DC constant is cut

A practical example of dismissing the constant in financial markets:
We can look at a low priced stock like GE, and a high priced stock like AAPL,
but when we compare daily returns between them, we don't care
what the long term price level is (the constant), we only care about daily percentage movements.

REFERENCE: PHASE SHIFTS

In time, a shift in peaks is called a lag (or lead for forward in time shift)
In frequency, that same shift is referred to as a "phase shift."

notes from:

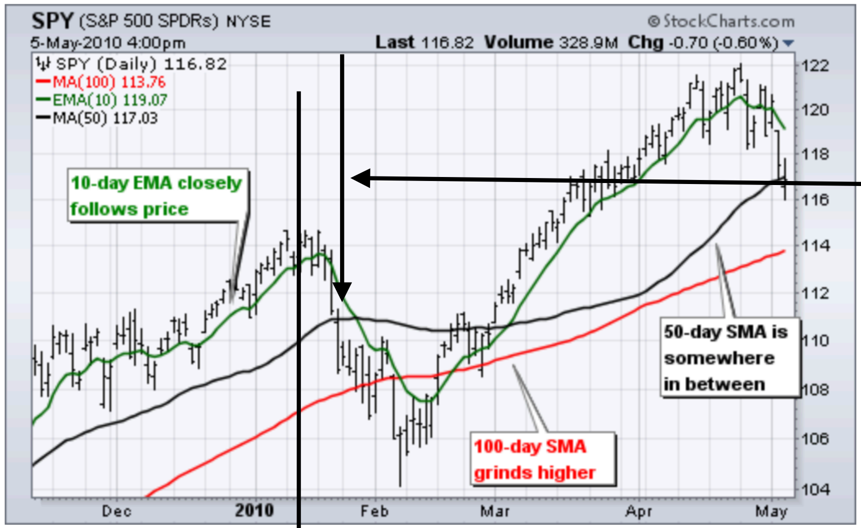
https://school.stockcharts.com/doku.php?id=technical_indicators:moving_averages

The Lag Factor

Moving average peak shift example.

The longer the moving average, the more the lag. A 10-day exponential moving average will hug prices quite closely and turn shortly after prices turn. Short moving averages are like speedboats - nimble and quick to change. In contrast, a 100-day moving average contains lots of past data that slows it down. Longer moving averages are like ocean tankers - lethargic and slow to change. It takes a larger and longer price movement for a 100-day moving average to change course.

(fractional) differencing and integration cause similar shifts in peaks



shift in peak of raw data versus 50 day moving avg curve

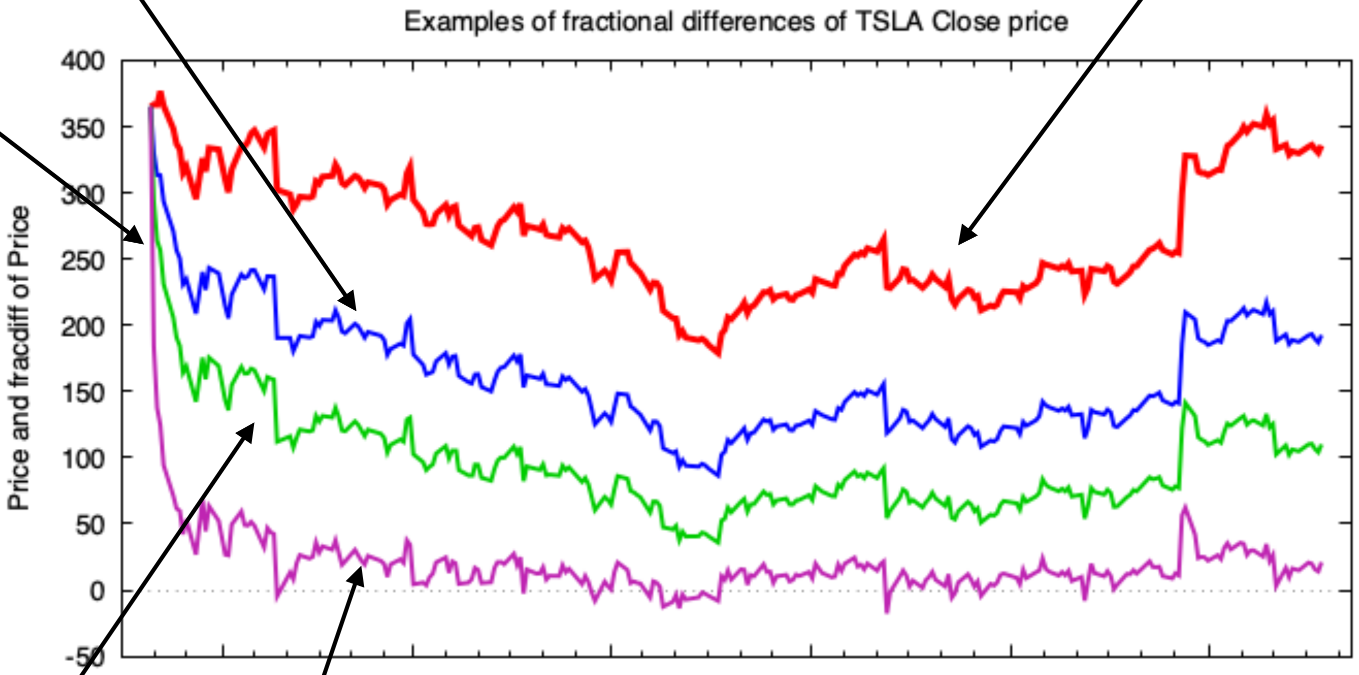
Click here for a live version of the chart.

The chart above shows the S&P 500 ETF with a 10-day EMA closely following prices and a 100-day SMA grinding higher. Even with the January-February decline, the 100-day SMA held the course and did not turn down. The 50-day SMA fits somewhere between the 10- and 100-day moving averages when it comes to the lag factor.

approx peak of raw data

Example:

fracdiff in GRETL applied to TSLA stock price series



fracdiff 0.1

original price

"meh" don't forget to throw out the bad points if you do further processing

bad points resulting from running out of prior data to apply the dot product to

fracdiff 0.2

fracdiff 0.5

except for the big blip* at start of curve

approaching stationary?
should do formal tests to check
see "ADF test for stationarity" in other references

***technical term**